06

OpenCV 사용하기 IV

Hough Transform및 차선 인식하기

06 Hough Transform

- **학습 목표: 이미지에서 직선을 찾는 허프 변환 알고리즘을 통해 직선 검출하기**
- **메인 코드:**

In [1]:
```
import cv2
import numpy as np


def nothing(x):
    pass

def region_of_interest(img, roi):
    vertices = np.array([roi], np.int32)
    mask = np.zeros_like(img)

    match_mask_color = 255
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image


def image_hough():
    road_img = cv2.imread("image/road_00.JPG")

    height = road_img.shape[0]
    width = road_img.shape[1]

    blur = cv2.GaussianBlur(road_img, (5, 5), 3)

    hsv = cv2.cvtColor(road_img, cv2.COLOR_BGR2HSV)

    roi = [
        (0, height),
        (width * 2 / 5, height * 5 / 7),
        (width * 3 / 5, height * 5 / 7),
        (width, height)
    ]
```

06 Hough Transform

- **코드 설명:**

```
    roi = [
        (0, height),
        (width * 2 / 5, height * 5 / 7),
        (width * 3 / 5, height * 5 / 7),
        (width, height)
    ]

    low_val = (0, 0, 190)
    high_val = (112, 255, 255)

    mask = cv2.inRange(hsv, low_val, high_val)
    cv2.imshow("mask", mask)
    roi_mask = region_of_interest(mask, roi)
    cv2.imshow("roi_mask", roi_mask)
    edges = cv2.Canny(roi_mask, 75, 150)

    rho = 1
    angle = np.pi / 180
    min_threshold = 60

    line_segments = cv2.HoughLinesP(edges, rho, angle, min_threshold,
                                    np.array([]), minLineLength=250, maxLineGap=200)


    n = 1
    for i in line_segments:
        print("{0} 번째 라인 : {1}".format(n, i))
        for j in i:
            print("x1 = {0} y1 = {1}₩nx2 = {2} y2 = {3}".format(j[0], j[1], j[2], j[3]))
            cv2.line(road_img, (j[0], j[1]), (j[2], j[3]), (255,0,0), 3)
        n += 1

    cv2.imshow("Edges", edges)
    cv2.imshow("Original", road_img)
    cv2.waitKey(0)

def dynamic_hough():
    road_img = cv2.imread("image/road_00.JPG")

    cv2.imshow("Original", road_img)
```

06 Hough Transform

- **코드 설명:**

```
def dynamic_hough():
    road_img = cv2.imread("image/road_00.JPG")

    cv2.imshow("Original", road_img)

    height = road_img.shape[0]
    width = road_img.shape[1]

    hsv = cv2.cvtColor(road_img, cv2.COLOR_BGR2HSV)

    roi = [
        (0, height),
        (width * 2 / 5, height * 5 / 7),
        (width * 3 / 5, height * 5 / 7),
        (width, height)
    ]

    low_val = (0, 0, 190)
    high_val = (112, 255, 255)

    mask = cv2.inRange(hsv, low_val, high_val)
    mask = region_of_interest(mask, roi)
    edges = cv2.Canny(mask, 75, 150)

    cv2.namedWindow('Control')
    cv2.createTrackbar('Treshold', 'Control', 0, 300, nothing)
    cv2.createTrackbar('minLineLength', 'Control', 0, 500, nothing)
    cv2.createTrackbar('maxLineGap', 'Control', 0, 500, nothing)

    cv2.setTrackbarPos('Treshold', 'Control', 10)
    cv2.setTrackbarPos('minLineLength', 'Control', 8)
    cv2.setTrackbarPos('maxLineGap', 'Control', 5)


    while True:

        rho = 1  # distance precision in pixel
        angle = np.pi / 180
        min_threshold = cv2.getTrackbarPos('Treshold', 'Control')
        minLength = cv2.getTrackbarPos('minLineLength', 'Control')
        maxGap = cv2.getTrackbarPos('maxLineGap', 'Control')
```

06 Hough Transform

- **코드 설명:**

```
    while True:

        rho = 1  # distance precision in pixel
        angle = np.pi / 180
        min_threshold = cv2.getTrackbarPos('Treshold', 'Control')
        minLength = cv2.getTrackbarPos('minLineLength', 'Control')
        maxGap = cv2.getTrackbarPos('maxLineGap', 'Control')

        line_segments = cv2.HoughLinesP(edges, rho, angle, min_threshold,
                                        np.array([]), minLineLength=minLength, maxLineGap=maxGap)

        canvas = np.zeros_like(road_img)
        n = 1
        for i in line_segments:
            print("{0} 번째 라인 : {1}".format(n, i))
            for j in i:
                print("x1 = {0} y1 = {1}₩nx2 = {2} y2 = {3}".format(j[0], j[1], j[2], j[3]))
                cv2.line(canvas, (j[0], j[1]), (j[2], j[3]), (255, 255, 0), 3)
            n += 1

        cv2.imshow("Line", canvas)
        cv2.imshow("Edges", edges)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cv2.destroyAllWindows()

if __name__ == "__main__":
    image_hough()
```

06 Hough Transform

- **결과:**


```
1 번째 라인 : [[238 721 567 516]]
x1 = 238 y1 = 721
x2 = 567 y2 = 516
2 번째 라인 : [[ 963  594 1216  717]]
x1 = 963 y1 = 594
x2 = 1216 y2 = 717
3 번째 라인 : [[241 719 566 516]]
x1 = 241 y1 = 719
x2 = 566 y2 = 516
```

- **코드설명:**

1. cv2.HoughLinesP (검출 이미지, rho, angle, minLineLength, maxLineGap)

rho는 거리 단위이며 픽셀을 의미합니다. 1로 놓으시면 됩니다.

angle은 라디안을 사용하며 0 ~ 180의 범위를 갖습니다. np.pi / 180 으로 놓으시면 됩니다.

threshold는 만나는 점의 개수입니다. 적당한 범위의 수를 주면 됩니다. 60으로 한번 설정해보겠습니다.

minLineLength는 선의 최소 길이를 의미하며 이보다 낮은 경우 무시합니다.
이 값은 너무 작게 주어선 안되며 최대한 크게 줘야합니다. 그러면 차선에서 점선으로 이뤄진 차선도 한 직선으로 검출되기 때문입니다. 하지만 반대로 너무 크게 줘버리면 양 차선 구분을 못한 엉뚱한 직선이 검출되기 때문에 최소길이 중 최대의 값을 줘야 합니다.

여기에서는 250을 놓도록 하겠습니다.

maxLineGap은 점들 사이의 최대 길이입니다. 마찬가지로 적당한 길이를 넣으면 됩니다.
너무 크거나 작으면 에러를 야기합니다. 이번 코드에서는 200을 놓도록 하겠습니다.

06 문제 5. 차선 인식하기

- **문제:**

  지금까지 학습한 것을 토대로 도로 영상에서 노란 차선과 흰 차선을 찾아서 허프 변환을 이용해 선분 좌표를 구하여 이를 원본 영상에 파란색 선으로 그려주세요.

- **결과:**